2023 – PARADIGMAS DE PROGRAMAÇÃO (MCTA016-13)

Turmas: DA1MCTA016-13SA e NA1MCTA016-13SA
Professores: Emilio Francesquini e Mario Leston Rey
E-mail: e.francesquini@ufabc.edu.br e mario.leston@ufabc.edu.br

1 Avisos

- [2023-05-09 Tue] Página da disciplina no ar

2 Informações Gerais

Esse oferecimento da disciplina será completamente **presencial**, incluindo aulas, plantões e avaliações. Contudo, o professor poderá responder a eventuais questionamentos e dúvidas no servidor do Discord no canal da disciplina cujo endereço está disponível abaixo.

Note

Todo o material das aulas é disponibilizado online segundo a Licença Creative Commons Atribuição-NãoComercial 4.0 Internacional (CC-BY-NC).

Tip

Inscreva-se o quanto antes no servidor do Discord da disciplina (https://discord.gg/JSgnfdE). Todos os anúncios e comunicações serão feitos por lá.

Figure 1: Servidor do Discord da disciplina.

Toda semana teremos uma aula de teoria e outra de prática, onde os conceitos vistos serão explorados. Lembramos também que a disciplina é unificada entre as turmas (diurna e noturna). Ou seja, os mesmos trabalhos, mesmos critérios, mesmas aulas entre todas as turmas. Então fique a vontade para escolher dentre os horários qual o melhor te atende, independente da turma onde estiver matriculado (é claro que, nas turmas práticas onde há uma limitação de espaço físico, os alunos matriculados naquele horário terão prioridade).

Quinzenamente serão disponibilizadas listas de exercícios sobre o conteúdo apresentado. As entregas destas listas serão consideradas como uma avaliação e deverão ser feitas em até duas semanas após a sua disponibilização. As notas dessas atividades serão utilizadas para a composição da média final. Teremos também duas avaliações escritas e presenciais e um projeto a ser entregue ao final da disciplina. A nota final será uma composição destas três avaliações.

2.1 Horário e local das aulas

Turma	Prof. Teoria	Prof Prática	Quarta	Sala		Sexta	Sala
DA1MCTA016-13SA	Emilio	Emilio	10h-12h	S-214-0		08h-10h	404-2
DA2MCTA016-13SA	Emilio	Mario	10h-12h	S-214-0		08h-10h	407-2
NA1MCTA016-13SA	Emilio	Emilio	21h-23h	S-214-0		19h-21h	404-2
NA2MCTA016-13SA	Emilio	Mario	21h-23h	S-214-0		19h-21h	407-2

2.2 Atendimento

- **Presencial**
  - Nos horários listados abaixo não é preciso confirmar ou marcar, apenas apareça! :-)
  - **Quarta-feira, das 19:00 às 21:00, Sala 509-2.**
  - **Sexta-feira, das 10:00 às 12:00, Sala 509-2.**
  - **Agendado por e-mail**
    - **Verifique minha agenda** e sugira pelo menos dois possíveis horários!
  - **Em sala de aula** - Após as aulas
- **Online**
  - Por e-mail.
  - Pelo Discord.

Lembramos também que a disciplina é unificada entre as turmas (diurna e noturna). Ou seja, os mesmos trabalhos, mesmos critérios, mesmas aulas entre todas as turmas. Então fique a vontade para escolher dentre os horários acima qual o melhor te atende, independente da turma onde estiver matriculado.

---

3 Sobre a Disciplina

MCTA016-13 - Paradigmas de Programação

- TPI: 2-2-4
- Recomendação: Processamento da Informação, Programação Orientada a Objetos

> **Objetivos**
>
> Esta disciplina traz à atenção do aluno as diversas diferenças fundamentais entregrandes famílias de linguagens de programação, tanto em teoria como de forma prática. Estavisão tem efeitos importantes, nem sempre perceptíveis: ao apreciar diversas técnicas deprogramação e mecanismos peculiares de linguagens de programação diferentes, o estudanteexpande seu leque de técnicas e rompe sua rigidez de concepção a respeito do que vem a serprogramar. Além disso, há situações onde um paradigma se aplicará com mais sucesso do queoutro. Pode-se sem dúvida afirmar que a programação em diferentes paradigmas auxilia namelhoria da qualidade da programação de forma geral.
>
> **Conteúdo Programático**
>
> Visão comparativa entre os paradigmas de programação. Paradigma funcional. Paradigma concorrente.

Fonte: Projeto Pedagógico do BCC 2017

---

4 Datas Importantes

4.1 Avaliações escritas

- Prova 1: 05/07
- Prova 2: 18/08

4.2 Listas quinzenais

Teremos listas quinzenais que serão utilizadas para a avaliação. Cada lista é composta de um número variado (não muito grande) de exercícios de programação:

- Os exercícios quinzenais de programação serão divididos em: 50% nível fácil, 25% nível intermediário e 25% nível desafiador

As datas de liberação dos enunciados e prazo para entrega são listados abaixo:

Lista	Enunciado/Prazo	Link (Matriculados e ouvintes)
1	02/06 - 19/06	
2	16/06 - 03/07	
3	30/06 - 17/07	
4	14/07 - 31/07	
5	28/07 - 14/08	
Bônus*	11/08 - **21/08**	

*Não serão aceitas entregas com atraso para a lista Bônus.

As playlists com a resolução de todas as listas dadas nos oferecimentos anteriores da disciplina podem ser encontradas aqui:

- 2021
- 2020
- 2022 (apenas exercícios, sem resolução)

- Todas as entregas devem ser feitas **exclusivamente pelo Github** Classroom através dos links acima. Listas enviadas por outros meios não serão aceitas.
- Entrega toda segunda-feira até às 23:59. Listas entregues com atraso de até um dia serão aceitas, contudo com um desconto de 50% da nota.

- O código submetido **precisa** compilar ao executar `stack test`.
- Caso não consiga fazer algum exercício coloque a palavra-chave `undefined` como definição da função. Ex.: `isEven = undefined`
- A execução de `stack test` não pode travar e nem entrar em loop infinito.

A falha em qualquer um desses casos acarretará em nota 0 na lista inteira. Assistam os vídeos com as correções das listas de exercício (em especial das listas 1 e 2) dos oferecimentos anteriores da disciplina que podem ser encontrados nos links listados acima.

4.3 Projeto de programação

- Proposta de projeto (Lembre-se, é apenas uma sugestão! Você é livre para escolher fazer o projeto que quiser!

Data para entrega: **21/08/2023 23h59**

- Entregas via Github Classroom. (link será disponibilizado em breve)
- O projeto pode ser feito em grupos de no máximo 3 pessoas
    - Apenas um dos integrantes do grupo deve fazer o upload do projeto para o GitHub e deve estar bem claro quais são os integrantes da equipe no relatório.
- Todos os *commits* e o *push* final devem ter sido feitos até no máximo às **23h59 do dia 21/08/2023**.
    - Atenção, caso ocorram/commits/pushes após o prazo, será considerada apenas a última versão para avaliação. Caso o último esteja com atraso > 3 dias, então será considerado o último commit até 3 dias (valendo no máximo 5).
    - Entregas em atraso sofrerão descontos conforme a tabela abaixo.

Dias em atraso	Nota máxima
1 dia	7
2 dias	6
3 dias	5
>3 dias	0

- No mesmo repositório do seu código, deverá haver um relatório de no máximo 1 páginas (em PDF! nada de `.md`, `.docx`, `.odt`, ou qualquer outro formato da moda...) descrevendo:
    - Qual é o seu projeto
    - Como utilizar (se for qualquer coisa diferente de `stack run`) o seu código

    - Dificuldades, surpresas e destaques do seu código
- O seu relatório também deverá conter **obrigatoriamente** um link para um vídeo que:
    - Esteja abrigado no Youtube (ou qualquer serviço semelhante)
    - Seja privado/não listado e disponível apenas àqueles com o link
    - Tenha no máximo 3 minutos (ou seja, <= 180 segundos). **Sem exceções**.
    - Mostre a compilação, o uso e uma apresentação do código do projeto
- Durante a correção o professor pode pedir por esclarecimentos adicionais.

**Que tipos de projetos serão aceitos? **

Lembre-se que você terá o quadrimestre todo para entregar o seu projeto. Logo é esperado que o seu projeto reflita um esforço de desenvolvimento compatível com o prazo disponível.

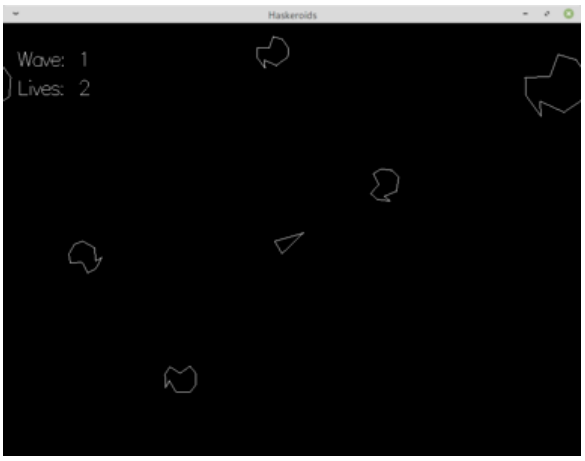
Como exemplo, em oferecimentos anteriores da disciplina tivemos alunos que implementaram jogos (inclusive dois deles que implementaram um clone completo do Pac-Man), algoritmos de IA e aprendizado de máquina e avaliações de desempenho, paralelizações de algoritmos clássicos, ferramentas completas para sincronização de arquivos, clientes de e-mail, servidores de banco de dados com suporte para SQL,...

Caso esteja com dúvidas se a sua ideia é o suficiente para um projeto final que será bem avaliado converse com o professor o quanto antes!

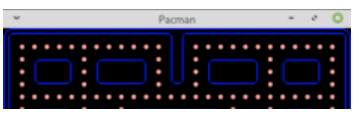
---

5 Projetos de exemplo de oferecimentos anteriores

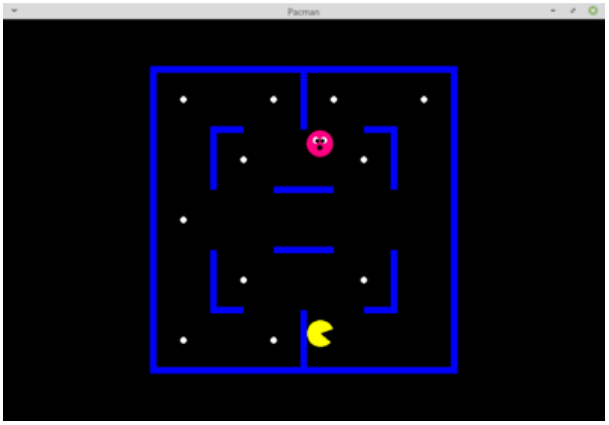
- **Haskeroid** - Por **Artur Henrique Allen Santos**
    - Código: haskeroid.zip



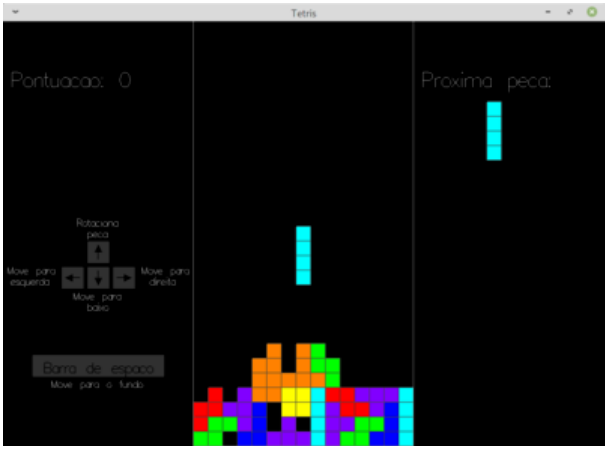
- **Pac-Man** - Por **Edson Gomes Martinelli e Rafael Akio Shishito Matos**
    - Código: pacman.zip



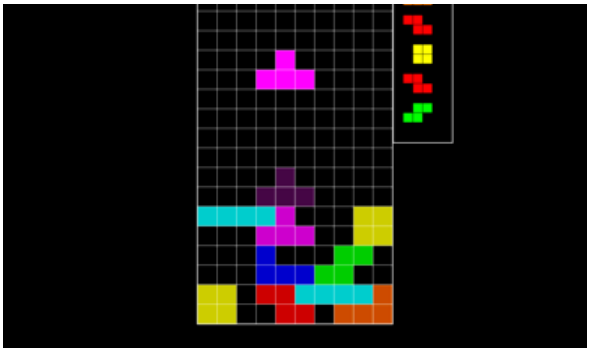
- **Pac-Man: Get the Curry** - Por **Jair Edipo Jerônimo e Michelle Kaori Hamada**
    - Código: pacman_curry.zip



- **Tetris I** - Por **Bárbara Dias de Sena**
    - Código: tetris_i.zip



- **Tetris II** - Por **Fabio Luis Arruda Fernandes**
    - Código: tetris_ii.zip



- **Xadrez** - Por **Eduardo Castilho Ferreira**
    - Código: xadrez.zip

6 Listas de Exercícios Complementares

Listas de exercícios extras (não valem nota) para aqueles que quiserem ir um pouco além...

- Lista 1
- Lista 2
- Lista 3
- Lista 4
- Lista 5
- Lista 6

7 Aulas

A disciplina será completamente presencial. Contudo, o material da disciplina mantém uma boa interseção com os oferecimentos anteriores da disciplina que foram online. Os vídeos e material escrito destas aulas está disponível online no seguinte endereço:

http://haskell.pesquisa.ufabc.edu.br/haskell

A playlist com todos os vídeos das aulas teóricas do curso pode ser encontrada aqui:

https://www.youtube.com/playlist?list=PLYItvall0TqJ25sVTLcMhxsE0Hci58mpQ

Para facilitar, na tabela abaixo você encontra os assuntos juntamente com links diretos para o conteúdo e vídeos.

Semana	Tópico	Material de Estudo	Playlist
1	Preparando o ambiente	https://haskell.pesquisa.ufabc.edu.br/haskell/00.bootstraping/	https://www.youtube.com/playlist?list=PLYItvall0TqLedblNsncIUfk3cHv_FS7O
1	Paradigmas de programação	https://haskell.pesquisa.ufabc.edu.br/haskell/01.paradigmas/	https://www.youtube.com/playlist?list=PLYItvall0TqIZ_ih1mSoc1roCZGIfj8-t
2	Cálculo Lambda	https://haskell.pesquisa.ufabc.edu.br/haskell/02.lambda/	https://www.youtube.com/playlist?list=PLYItvall0TqKPbnSblJ_fxNIFRgEoI-7_
2	Conceitos Básicos - Parte 1	https://haskell.pesquisa.ufabc.edu.br/haskell/03.haskell.basico.1/	https://www.youtube.com/playlist?list=PLYItvall0TqLlCPN9vbDIc8FAKhG-RfbM
2	Conceitos Básicos - Parte 2	https://haskell.pesquisa.ufabc.edu.br/haskell/04.haskell.basico.2/	https://www.youtube.com/playlist?list=PLYItvall0TqLlCPN9vbDIc8FAKhG-RfbM
3	QuickCheck	https://haskell.pesquisa.ufabc.edu.br/haskell/05.quickcheck/	https://www.youtube.com/playlist?list=PLYItvall0TqJ25sVTLcMhxsE0Hci58mpQ
3	Funções de Alta Ordem	https://haskell.pesquisa.ufabc.edu.br/haskell/06.higher.order/	https://www.youtube.com/playlist?list=PLYItvall0TqLBLt6oXFVBaloU7-xZsV-v
4	Tipos de Dados Algébricos	https://haskell.pesquisa.ufabc.edu.br/haskell/07.adt/	https://www.youtube.com/playlist?list=PLYItvall0TqJWdfiLuMsIm_4ag3fJjHTo
5	Monoid e Foldable	https://haskell.pesquisa.ufabc.edu.br/haskell/08.monoids.foldable/	https://www.youtube.com/playlist?list=PLYItvall0TqIzShhLgcVVti0d0kAaJMWC
5	Functor	https://haskell.pesquisa.ufabc.edu.br/haskell/09.functors/	https://www.youtube.com/playlist?list=PLYItvall0TqKtSdeBAjz0GOM_3oq3pP8-
6	Applicative Functor e Traversable	https://haskell.pesquisa.ufabc.edu.br/haskell/10.applicatives.traverse/	https://www.youtube.com/playlist?list=PLYItvall0TqLyKtl_Cnx0g_H4Bk686e4Y
7	Monads - Parte 1	https://haskell.pesquisa.ufabc.edu.br/haskell/11.monads/	https://www.youtube.com/playlist?list=PLYItvall0TqLW_mPtIpVA8qHpuQ3YbnVO
8	Monads - Parte 1	https://haskell.pesquisa.ufabc.edu.br/haskell/11.monads/	https://www.youtube.com/playlist?list=PLYItvall0TqIl6y3FydOuEjFMH6jp7ROm
9	Avaliação Preguiçosa	https://haskell.pesquisa.ufabc.edu.br/haskell/12.laziness/	https://www.youtube.com/playlist?list=PLYItvall0TqJwLa9rY-bT_B9-EmtaiPT0
9	Haskell Paralelo	https://haskell.pesquisa.ufabc.edu.br/haskell/13.paralelismo/	https://www.youtube.com/playlist?list=PLYItvall0TqJ1jteUbGOHfBycg5NM9thq
10	Programação Concorrente	https://haskell.pesquisa.ufabc.edu.br/haskell/14.concorrencia/	https://www.youtube.com/playlist?list=PLYItvall0TqKz0Jw8RTA2epq8VimzSqGp

Semana	Tópico	Material de Estudo	Playlist
10	Estruturas de dados funcionais e persistência	https://haskell.pesquisa.ufabc.edu.br/estruturas-de-dados//01.persistencia/	https://www.youtube.com/playlist?list=PLYItvall0TqIOxQzCMsK3zciIXxxgzlcG
11	Árvores: Roseiras e Rubro-Negras	https://haskell.pesquisa.ufabc.edu.br/estruturas-de-dados//02.arvores/	https://youtube.com/playlist?list=PLYItvall0TqLE8OELzykHZOn7LaKu6-Ze
12	Heaps funcionais	https://haskell.pesquisa.ufabc.edu.br/estruturas-de-dados//03.heaps/	

8 Critério de avaliação

Honestidade Acadêmica

Entre outros, o código de ética da UFABC estabelece em seu artigo 25 que é **eticamente inaceitável** que os discentes:
I - fraudem avaliações;
II - fabriquem ou falsifiquem dados;
III - plagiem ou não creditem devidamente autoria;
IV - aceitem autoria de material academico sem participação na produção;
V - vendam ou cedam autoria de material acadêmico próprio a pessoas que não participaram da produção.

Muitos ainda têm dúvidas sobre a interpretação das regras definidas pelo Código de Ética da UFABC. Por esta razão, diversos professores elaboraram um documento (**disponível aqui**) com vários exemplos e esclarecendo a interpretação das regras acima. Abaixo uma versão resumida. **Sempre consulte o documento completo ou converse com o seu professor em caso de dúvidas!**

- **Regra 1** - Você não pode enviar para avaliação um trabalho que não seja de sua própria autoria ou que seja derivado/baseado em soluções elaboradas por outros.
- **Regra 2** - Você não pode compartilhar a sua solução com outros alunos nem pedir aos seus colegas que compartilhem as soluções deles com você.
- **Regra 3** - Nos trabalhos enviados para avaliação você deve indicar eventuais assistências que você tenha recebido.

  ATENÇÃO: todos os trabalhos enviados para avaliação poderão ser verificados por um sistema automatizado de detecção de plágio.

Qualquer violação às regras descritas acima implicará:

- Descarte dos conceitos atribuídos a TODAS as tarefas avaliativas regulares de TODOS os envolvidos, causando assim suas reprovações automáticas com conceito F .
- Possível denúncia à Comissão de Transgressões Disciplinares Discentes da Graduação, a qual decidirá sobre a punição adequada à violação que pode resultar em advertência, suspensão ou desligamento , de acordo com os artigos 78-82 do Regimento Geral da UFABC.
- Possível denúncia apresentada à Comissão de Ética da UFABC, de acordo com o artigo 25 do Código de Ética da UFABC.

A avaliação da disciplina será composta pelas seguintes notas:

- N_a é a nota do projeto
- N_P é a nota do projeto
- N_L é a nota das listas de exercícios semanais

A nota N_L será calculada como a média aritmética das listas quinzenais, cada lista valendo 10 pontos, descartando-se a lista com a pior nota. Ou seja, se por qualquer motivo você não puder enviar uma das listas ou caso tenha sido avaliada como uma nota baixa, basta que faça todas as demais para que não haja prejuízo na sua nota.

A nota das avaliações escritas N_a será calculada como a média ponderada da primeira e segunda avaliações (P_1 e P_2) com pesos 2 e 3 respectivamente:

$$N_a = \frac{2P_1 + 3P_2}{5}$$

A nota final (N_F) será determinada pela **média harmônica ponderada ** de N_a, N_P e N_L, com pesos 5, 3 e 2 respectivamente:

$$N_F = \frac{10}{\frac{5}{\max\{0.1, N_a\}} + \frac{3}{\max\{0.1, N_P\}} + \frac{2}{\max\{0.1, N_L\}}}$$

O conceito final (C_F) será obtido de acordo com a equação abaixo:

$$C_F = \begin{cases} \mathbf{F}, & \text{se } N_F \in [0,0; 5,0) \\ \mathbf{D}, & \text{se } N_F \in [5,0; 6,0) \\ \mathbf{C}, & \text{se } N_F \in [6,0; 7,0) \\ \mathbf{B}, & \text{se } N_F \in [7,0; 8,5) \\ \mathbf{A}, & \text{se } N_F \in [8,5; 10,0] \end{cases}$$

Warning

Caso seja verificado ocorrência de plágio no projeto final, o aluno será automaticamente reprovado com F.

8.1 Recuperação

A resolução ConsEPE nº 182 assegura a todos os alunos de graduação com C_F igual a **D** ou **F** o direito a fazer uso de mecanismos de recuperação.

A recuperação será feita através de uma nova avaliação escrita a ser marcada entre os dias 18/09/2023 e 30/09/2023. A sua nota será utilizada para compor a o conceito pós-recuperação C_R conforme as equações abaixo:

$$N_R = \frac{P_R + N_F}{2}$$

Caso 1 $C_F = D$**:**

$$C_R = \begin{cases} \mathbf{C}, & \text{se } N_R \geq 6,0 \\ \mathbf{D}, & \text{caso contrário} \end{cases}$$

Caso 2 $C_F = F$:

$$C_R = \begin{cases} \mathbf{D}, & \text{se } N_R \geq 5,0 \\ \mathbf{F}, & \text{caso contrário} \end{cases}$$

8.2 Regulamentações Relevantes

- Resolução ConsEPE nº 182 - Regulamenta a aplicação de mecanismos de recuperação
- Código de Ética da UFABC
- Resolução ConsUni nº 63 - Regimento Geral da UFABC

---

9 Recursos Online

9.1 Grupos, listas, páginas, ...

- Grupo de Estudos em Haskell da UFABC
  - **Não esqueça de participar do Discord**: https://discord.gg/JSgnfdE
- Haskell Home Page
- #haskell IRC channel
- StackOverflow
- A lista Haskell-beginners é um bom lugar para buscar respostas para perguntas básicas
- A lista Haskell-cafe é boa para buscar respostas a perguntas mais elaboradas
- O Haskell Wiki tem uma lista de respostas às perguntas mais comuns

9.2 Disciplinas em Haskell

- MCTA016-13: Paradigmas de Programação (em Haskell), UFABC. 2022, 2021, 2020, 2019, 2018.
- CR062-Programação Funcional em Haskell, UFABC. 2019, 2018.
- Estruturas de Dados Puramente Funcionais. 2019.
- G51PGP: Programming Paradigms, University of Nottingham. 2019.
- CS653: Functional Programming, Indian Institute of Technology Kanpur. 2018.
- CIS 194: Introduction to Haskell, University of Pennsylvania. 2016, 2015, 2014, 2013.
- ... mais exemplos aqui

9.3 Leituras

- Haskell Wikibook
- A Gentle Introduction to Haskell
- Haskell Cheat Sheet
- What I Wish I Knew When Learning Haskell

9.4 Documentação

- Standard library documentation
- Hackage - Repositório de pacotes
- Hoogle - Procurando funções para as quais você não sabe o nome?
- Hayoo - Parecido com o Hoogle, porém mais completo (procura todo o Hackage, mas está frequentemente fora

do ar...)
- Se você realmente precisar, a linguagem e a biblioteca padrão do Haskell são definidos pelo Haskell 2010 - Language Report

---

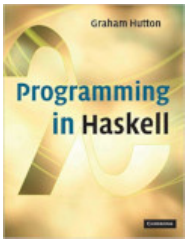
10 Bibliografia

Os principal texto utilizado neste curso será o **GH Segunda Edição**.

[GH]

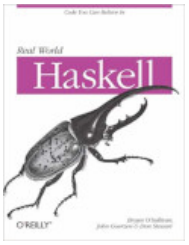
- ***Programming in Haskell***. 2nd Edition.
  - Por *Graham Hutton*.

A primeira edição, que tem boa parte do conteúdo da segunda edição, está disponível na biblioteca:



1st Edition (antiga)

- Link Biblioteca: http://biblioteca.ufabc.edu.br/index.php?codigo_sophia=15287

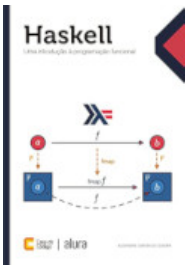


[SGS]

- *Real World Haskell*.
  - Por *Bryan O'Sullivan*, *John Goerzen* e *Don Stewart*.
  - Disponível **gratuitamente** em: http://book.realworldhaskell.org/

[ML]

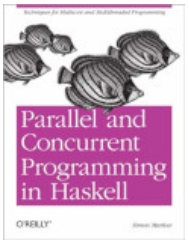
- *Learn You a Haskell for Great Good!: A Beginner's Guide*.
  - Por *Miran Lipovača*.
  - Disponível **gratuitamente** em: http://learnyouahaskell.com/



[HIPF]

- *Haskell: Uma introdução à programação funcional*
  - Por *Alexandre Garcia de Oliveira*
  - Disponível em: https://www.casadocodigo.com.br/products/livro-haskell?_pos=2&_sid=8ab00a083&_ss=r

---



[SM]

- *Parallel and Concurrent Programming in Haskell: Techniques for Multicore and Multithreaded Programming*.
  - Por *Simon Marlow*.
  - Disponível **gratuitamente** em: https://www.oreilly.com/library/view/parallel-and-concurrent/9781449335939/